# Лабораторная работа: Интеграция моделей машинного обучения в Django приложение

## Цель работы:

Научиться интегрировать обученные модели машинного обучения в веб-приложение на базе Django, предоставляя API или веб-интерфейс для использования модели.

## Этапы выполнения работы

### 1. Подготовка окружения

1. Установите Python и виртуальное окружение:
 ```bash
 python3 -m venv venv
 source venv/bin/activate # Linux/Mac
 venv\Scripts\activate # Windows
 ```
2. Установите Django и зависимости для ML:
 ```bash
 pip install django pandas numpy scikit-learn joblib
 ```

### 2. Создание Django-приложения

1. Создайте новый Django-проект:
 ```bash
 django-admin startproject ml\_project
 cd ml\_project
 python manage.py startapp ml\_app
 ```
2. Добавьте приложение `ml\_app` в `INSTALLED\_APPS` в файле `settings.py`.

### 3. Обучение и сохранение модели

1. Создайте скрипт для обучения модели, например, `train\_model.py`:
 ```python
 import pandas as pd
 from sklearn.model\_selection import train\_test\_split
 from sklearn.ensemble import RandomForestClassifier
 from sklearn.metrics import accuracy\_score
 import joblib

 # Загружаем данные
 data = pd.read\_csv('data.csv')
 X = data.drop('target', axis=1)
 y = data['target']

 # Разделяем на обучающую и тестовую выборки
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

 # Обучаем модель
 model = RandomForestClassifier()
 model.fit(X\_train, y\_train)

 # Оцениваем модель
 predictions = model.predict(X\_test)
 print(f"Accuracy: {accuracy\_score(y\_test, predictions)}")

 # Сохраняем модель
 joblib.dump(model, 'ml\_app/model.pkl')
 ```
2. Убедитесь, что файл `model.pkl` находится в папке `ml\_app`.

### 4. Интеграция модели в Django

1. Создайте модуль для загрузки модели в `ml\_app/utils.py`:
 ```python
 import joblib
 import os

 def load\_model():
 model\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'model.pkl')
 return joblib.load(model\_path)
 ```

2. Настройте views для обработки запросов. В файле `ml\_app/views.py`:
 ```python
 from django.http import JsonResponse
 from .utils import load\_model
 import pandas as pd

 model = load\_model()

 def predict(request):
 try:
 # Получаем данные из запроса
 input\_data = request.GET.dict()
 df = pd.DataFrame([input\_data])

 # Предсказание
 prediction = model.predict(df)
 return JsonResponse({'prediction': prediction.tolist()})
 except Exception as e:
 return JsonResponse({'error': str(e)})
 ```

3. Настройте маршруты (URLs):
 В файле `ml\_app/urls.py`:
 ```python
 from django.urls import path
 from .views import predict

 urlpatterns = [
 path('predict/', predict, name='predict'),
 ]
 ```
 В файле `ml\_project/urls.py`:
 ```python
 from django.contrib import admin
 from django.urls import path, include

 urlpatterns = [
 path('admin/', admin.site.urls),
 path('ml/', include('ml\_app.urls')),
 ]
 ```

### 5. Тестирование API

1. Запустите сервер:
 ```bash
 python manage.py runserver
 ```
2. Воспользуйтесь браузером или Postman для тестирования. Пример GET-запроса:
 ```
 http://127.0.0.1:8000/ml/predict/?feature1=value1&feature2=value2
 ```

### 6. (Опционально) Создание веб-интерфейса

1. Добавьте шаблон HTML в папку `ml\_app/templates/ml\_app/`:
 ```html
 <!-- predict.html -->
 <form method="get" action="{% url 'predict' %}">
 <label>Feature 1:</label>
 <input type="text" name="feature1">
 <label>Feature 2:</label>
 <input type="text" name="feature2">
 <button type="submit">Predict</button>
 </form>
 ```
2. Настройте views для рендера шаблона:
 В файле `ml\_app/views.py`:
 ```python
 from django.shortcuts import render

 def form\_view(request):
 return render(request, 'ml\_app/predict.html')
 ```

3. Обновите маршруты:
 В файле `ml\_app/urls.py`:
 ```python
 urlpatterns = [
 path('', form\_view, name='form\_view'),
 path('predict/', predict, name='predict'),
 ]
 ```

### 7. Итог

Теперь у вас есть Django-приложение, которое может:
- Принимать запросы к модели через API.
- Предоставлять простой веб-интерфейс для ввода данных и получения предсказаний.